On the Identity Type as the Type of Computational Paths

Arthur F. Ramos[1]

Centro de Informtica
Universidade Federal de Pernambuco
Recife, Brazil

Ruy J. G. B. de Queiroz[2]

Centro de Informtica
Universidade Federal de Pernambuco
Recife, Brazil

Anjolina G. de Oliveira[3]

Centro de Informtica
Universidade Federal de Pernambuco
Recife, Brazil

Abstract

We introduce a new way of formalizing the intensional identity type based on the fact that a entity known as computational paths can be interpreted as terms of the identity type. Our approach enjoys the fact that our elimination rule is easy to understand and use. We make this point clear constructing terms of some relevant types using our proposed elimination rule. We also show that the identity type, as defined by our approach, induces a groupoid structure. This result is on par with the fact that the traditional identity type induces a groupoid, as exposed by Hofmann & Streicher (1994).

Keywords: Identity type, computational paths, equality theory, path-based constructions, type theory, groupoid model.

1 Introduction

One interesting peculiarity of Martin-Löf's Intensional Type Theory is the existence of two distinct kind of equalities between terms of the same type. The first one is

[1] Email: afr@cin.ufpe.br
[2] Email: ruy@cin.ufpe.br
[3] Email:ago@cin.ufpe.br

originated by the fact that equality can be seen as a type. The second one is originated by the fact that two terms can be equal by definition.

The treatment of an equality as a type gives rise to a extremely interesting type known as identity type. The idea is that, given terms a, b of a type A, one may form the type whose elements are proofs that a and b are equal elements of type A. This type is represented by $Id_A(a, b)$. A term $p : Id_A(a, b)$ makes up for the *grounds* [14] (or proof) that estabilishes that a is indeed equal to b. We say that a is *propositionally* equal to b.

The second kind of equality is called definitional and is denotated by $\equiv$. It occurs when two terms are equal by definition, i.e., there is no need for an evidence or a proof to estabilish the equality. A classic example, given in [1], is to consider any function definition, for example, $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) \equiv x^2$. For $x = 3$, we have, by definition, that $f(x) \equiv 3^2$. We are unable to conclude that $f(3) \equiv 9$ though. The problem is that, to conclude that $3^2 = 9$, we need additional evidence. We need a way to compute 3^2, i.e., we need to use exponentials (or multiplications) to conclude that 3^2 is equal to 9. That way, we can only prove that 3^2 is propositionally equal to 9, i.e., there is a $p : Id_{\mathbb{N}}(3^2, 9)$.

Between those two kind of equalities, the propositional one is, without a doubt, the most interesting one. This claim is based on the fact that many interesting results have been achieved using the identity type. One of these was the discovery of the Univalent Models in 2005 by Vladimir Voevodsky [16]. A groundbreaking result has arisen from Voevdsky's work: the connection between type theory and homotopy theory. The intuitive connection is simple: a term $a : A$ can be considered as a point of the space A and $p : Id_A(a, b)$ is a homotopical path between points $a, b \in A$ [1]. This has given rise to a whole new area of research, known as Homotopy Type Theory. It leads to a new perspective on the study of equality, as expressed by Voevodsky in a recent talk in *The Paul Bernays Lectures* (Sept 2014, Zurich): equality (for abstract sets) should be looked at as a *structure* rather than as a *relation*.

Motivated by the fact that the identity type has given rise to such interesting concepts, we have been engaged in a process of revisiting the construction of the intensional identity type, as originally proposed by Martin-Löf. Although beautifully defined, we have noticed that proofs that uses the identity type can be sometimes a little too complex. The elimination rule of the intensional identity type encapsulates lots of information, sometimes making too troublesome the process of finding the reason that builds the correct type.

Inspired by the path-based approach of the homotopic interpretation, we believe that a similar approach can be used to define the identity type in type theory. To achieve that, we have been using a notion of *computational paths*. The interpretation will be similar to the homotopic one: a term $p : Id_A(a, b)$ will be a computational path betweem terms $a, b : A$, and such path will be the result of a sequence of rewrites. In the sequel, we shall define formally the concept of a computational path. The main idea, i.e. proofs of equality statements as (reversible) sequences of rewrites, is not new, as it goes back to a paper entitled "Equality in labelled deductive systems and the functional interpretation of propositional equality", presented in December 1993 at the *9th Amsterdam Colloquium*, and published in the proceedings in 1994

[8].

2 Computational Paths

Since computational path is a generic term, it is important to emphasize the fact that we are using the term computational path in the sense defined by [6]. A computational path is based on the idea that it is possible to formally define when two computational objects $a, b : A$ are equal. These two objects are equal if one can reach b from a applying a sequence of axioms or rules. This sequence of operations forms a path. Since it is between two computational objects, it is said that this path is a computational one. Also, an application of a axiom or a rule transforms (or rewrite) an term in another. For that reason, a computational path is also known as a sequence of rewrites. Nevertheless, before we define formally a computational path, we can take a look at one famous equality theory, the $\lambda\beta\eta - equality$ [10]:

Definition 2.1 The $\lambda\beta\eta$-*equality* is composed by the following axioms:

(α) $\lambda x.M = \lambda y.[y/x]M \quad$ if $y \notin FV(M)$;

(β) $(\lambda x.M)N = [N/x]M$;

(ρ) $M = M$;

(η) $(\lambda x.Mx) = M \quad (x \notin FV(M))$.

And the following rules of inference:

$$(\mu) \quad \frac{M = M'}{NM = NM'} \qquad (\tau) \frac{M = N \qquad N = P}{M = P}$$

$$(\nu) \quad \frac{M = M'}{MN = M'N} \qquad (\sigma) \frac{M = N}{N = M}$$

$$(\xi) \quad \frac{M = M'}{\lambda x.M = \lambda x.M'}$$

Definition 2.2 ([10]) P is β-equal or β-convertible to Q (notation $P =_\beta Q$) iff Q is obtained from P by a finite (perhaps empty) series of β-contractions and reversed β-contractions and changes of bound variables. That is, $P =_\beta Q$ iff **there exist** $P_0, \ldots, P_n$ $(n \geq 0)$ such that $P_0 \equiv P$, $P_n \equiv Q$, $(\forall i \leq n-1)(P_i \triangleright_{1\beta} P_{i+1}$ or $P_{i+1} \triangleright_{1\beta} P_i$ or $P_i \equiv_\alpha P_{i+1})$.

(NB: equality with an **existential** force, which will show in the proof rules for the identity type.)

The same happens with $\lambda\beta\eta$-equality:

Definition 2.3 ($\lambda\beta\eta$-equality [10]) The equality-relation determined by the theory $\lambda\beta\eta$ is called $=_{\beta\eta}$; that is, we define

$$M =_{\beta\eta} N \quad \Leftrightarrow \quad \lambda\beta\eta \vdash M = N.$$

Example 2.4 Take the term $M \equiv (\lambda x.(\lambda y.yx)(\lambda w.zw))v$. Then, it is $\beta\eta$-equal to $N \equiv zv$ because of the sequence:

$(\lambda x.(\lambda y.yx)(\lambda w.zw))v, \quad (\lambda x.(\lambda y.yx)z)v, \quad (\lambda y.yv)z, \quad zv$
which starts from M and ends with N, and each member of the sequence is obtained via 1-step β- or η-contraction of a previous term in the sequence. To take this sequence into a *path*, one has to apply transitivity twice, as we do in the example 3.1 below.

The aforementioned theory estabilishes the equality between two λ-terms. Since we are working with computational objects as terms of a type, we need to translate the $\lambda\beta\eta$-equality to a suitable equality theory based on Martin Löf's type theory. We obtain:

Definition 2.5 The equality theory of Martin Löf's type theory has the following basic proof rules for the Π-type:

$$(\beta) \quad \frac{\begin{array}{cc}[x:A]\\ N:A & M:B\end{array}}{(\lambda x.M)N = M[N/x] : B} \qquad (\xi) \quad \frac{\begin{array}{c}[x:A]\\ M = M' : B\end{array}}{\lambda x.M = \lambda x.M' : (\Pi x:A)B}$$

$$(\rho) \quad \frac{M:A}{M = M : A} \qquad (\mu) \quad \frac{M = M' : A \qquad N : (\Pi x:A)B}{NM = NM' : B}$$

$$(\sigma) \quad \frac{M = N : A}{N = M : A} \qquad (\upsilon) \quad \frac{N : A \qquad M = M' : (\Pi x:A)B}{MN = M'N : B}$$

$$(\tau) \quad \frac{M = N : A \qquad N = P : A}{M = P : A}$$

$$(\eta) \quad \frac{M : (\Pi x:A)B}{(\lambda x.Mx) = M : (\Pi x:A)B} \; (x \notin FV(M))$$

We are finally able to formally define computational paths:

Definition 2.6 Let a and b be elements of a type A. Then, a *computational path* s from a to b is a sequence of rewrites (each rewrite is an application of the inference rules of the equality theory of type theory or is a change of bound variables). We denote that by $a =_s b$.

The definition makes clear the power of the computational paths. Since each rewrite is an application of intuitive axioms, working with computational paths is straightfoward. In addition, the presence of axioms such as the reflexivity (ρ), the transitivity (τ) and the symmetry(σ) opens the possibility of computational paths inducing a groupoidal structure. This will be the case, as we will see further in this work.

3 Identity Type

As we have already mentioned, our objective is to propose a formalization to the identity type using computational paths. We also want to make clear that we are working with the intensional version of the identity type. We assure that due to the

fact that there exists an extensional version. Nevertheless, this extensional version does not have most of the interesting properties that the intensional one has (the homotopic interpretation, for example). Since our approach is based on computational paths, we will sometimes refer to our formulation as the *path-based* approach and the traditional formulation as the *pathless* approach. Before the deductions that build the path-based identity type, we would like to show the construction of the traditional approach [9]:

$$\frac{A \text{ type} \qquad M : A \qquad N : A}{Id_A(M,N) \text{ type}}\, Id - F \qquad \frac{a : A}{r(a) : Id_A(a,a)}\, Id - I$$

$$\frac{M : A \qquad N : A \qquad P : Id_A(M,N) \qquad \begin{matrix}[x:A] \\ Q(x) : C(x,x,r(x))\end{matrix} \qquad \begin{matrix}[x:A, y:A, z:Id_A(x,y)] \\ C(x,y,z) \text{ type}\end{matrix}}{J(P,Q) : C(M,N,P)}\, Id - E$$

As we can see, the traditional approach is based on the assumption that there is only one basic proof, the reflexive one. Every proof is then based upon the reflexivity. The complexity of the equality expression does not matter. If two terms $a, b : A$ are really propositionally equal, the constructor J will construct a term $p : Id_A(a,b)$. It is incredible to think that a reflexive base is capable of building complexes proofs. Nevertheless, the beauty of the identity type comes with a setback: sometimes the constructor J is not easy to use. The problem is the high amount of information. To use J to build a type, one needs to come up with a suitable reason that justifies the equality, i.e., suitable $x, y : A$ and $z : Id_A(x,y)$ that build the correct $C(x,y,z)$. After obtaining the correct C, one sometimes realize that $C(x,x,r(x))$ is a type other than $Id_A(x,x)$. If $C(x,x,r(x)) \equiv Id_A(x,x)$, Q would be the trivial reflexive term, $r(x)$, and the construction would be complete. But if it is not the case, one will need to use the constructor J recursively to build the term Q. Of course it means that one will need to find yet another reason to build a suitable $C'(x,y,z)$. Since finding reasons can be cumbersome, proofs using J can be really difficult sometimes. It can get even worse: one may need lots of steps and applications of J before obtaning $C(x,x,r(x)) \equiv Id_A(x,x)$. On the other hand, after completing the proof, the correctness can be easily checked by computers. The real problem is how hard it is to find the correct reason and the number of recursive steps.

Simplifying this process is one of the motivations of the path-based approach. The path-based identity type will be constructed the same way that the pathless one had been, i.e., using natural deductions to formalize each rule. We start with the formation and introduction:X

$$\frac{A \text{ type} \qquad a : A \qquad b : A}{Id_A(a,b) \text{ type}}\, Id - F \qquad \frac{a =_s b : A}{s(a,b) : Id_A(a,b)}\, Id - I_1$$

No surprises about the $Id - F$, since it is equal to the pathless approach. The $Id - I_1$ is where lies the first difference: if we have a computational path between $a, b : A$, then the path is a evidence of the equality of these terms (since from a

we can apply a sequence of rewrites, represented by s, to reach b). Therefore, we introduce the term $s(a,b) : Id_A(a,b)$. As an example of how such a proof term is built up, let us recall a pair of terms used in a previous example:

Example 3.1 The term $M \equiv (\lambda x.(\lambda y.yx)(\lambda w.zw))v$ is $\beta\eta$-equal to $N \equiv zv$ because of the sequence:
$(\lambda x.(\lambda y.yx)(\lambda w.zw))v, \quad (\lambda x.(\lambda y.yx)z)v, \quad (\lambda y.yv)z, \quad zv$
Now, taking this sequence into a path leads us to the following:
The first is equal to the second based on the grounds:
$\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v)$
The second is equal to the third based on the grounds:
$\beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z)$
Now, the first is equal to the third based on the grounds:
$\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v), \beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z))$
Now, the third is equal to the fourth one based on the grounds:
$\beta((\lambda y.yv)z, zv)$
Thus, the first one is equal to the fourth one based on the grounds:
$\tau(\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v), \beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z)), \beta((\lambda y.yv)z, zv))).$

There is another introduction rule:

$$\frac{a =_s b : A \qquad a =_t b : A \qquad s =_z t : Id_A(a,b)}{s(a,b) =_{\xi(z)} t(a,b) : Id_A(a,b)}\, Id - I_2$$

This introduction brings about an interesting aspect, since it introduces an important construction: definitional equality between paths, leading to the possibility of constructing paths between paths. Since a computational path is also a computational object, it is natural to think that we can estabilish the equality of two paths. Since in our approach the evidence of equality is given by paths, then the equality of two paths must be estabilished by another path. If the paths are equal, then this rule indicates that the identity terms will be also equal. The existence of paths between paths is essential, since it turns possible the existence of higher structures (as we will see further in this work). The first elimination follows:

$$\frac{\begin{array}{cc} & [a =_g b : A] \\ m : Id_A(a,b) & h(g) : C \end{array}}{REWR(m, \acute{g}.h(g)) : C}\, Id - E_1$$

The elimination works as follows: if from the equality of a and b (estabilished by a path g) one can build a term $h(g) : C$, then from another proof of the equality of these two terms (estabilished by m) one also should be capable of building a term of type C. That idea is captured by the constructor $REWR$. It receives m and the $\acute{g}.h(g)$, where $\acute{g}$ is an abstraction. More specifically, $\acute{g}$ is an abstraction over the variable g, for which the main rules of conversion of λ-abstraction hold. To simulate the fact that from m one should be capable of building a term of C, one should be capable of applying m in the expression $\acute{g}.h(g)$, obtaining $h(m/g)$ (i.e.,

the expression h with the free occurrences of g replaced with the term m, so that $h(m) : C$. Of course, this computation should be formalized by a rule. That is exactly what we will do, but before that, let us s show the second elimination:

$$\frac{\begin{array}{cc} & [a =_g b : A] \\ p =_r q : Id_A(a,b) & h(g) : C \end{array}}{REWR(p, \acute{g}.h(g)) =_{\mu(r)} REWR(q, \acute{g}.h(g)) : C}\, Id - E_2$$

This rule is similar to what happened in the second introduction rule. If we have two equal paths (equality estabilished by another path), then if we apply the elimination in each of these paths we should obtain equal $REWR$ terms. Now, we define the computation rule that we have just mentioned:

$$\frac{\dfrac{a =_m b : A}{m(a,b) : Id_A(a,b)}\, Id - I_1 \qquad \begin{array}{c}[a =_g b : A] \\ h(g) : C\end{array}}{REWR(m, \acute{g}.h(g)) : C}\, Id - E_1 \quad \rhd_\beta \quad \begin{array}{c} a =_m b : A \\ h(m/g) : C \end{array}$$

This rules formalizes what we have described. It creates a reduction rule for the term $REWR(m, \acute{g}.h(g))$. This reduction is just the application of m in $\acute{g}.h(g)$, obtaining $h(m/g)$. Since this reduction is similar to the β one of the $\lambda\beta\eta$-equality, we chose to call it β-reduction for computational paths. There is another reduction rule:

$$\frac{\begin{array}{cc} & \dfrac{[a =_t b : A]}{t(a,b) : Id_A(a,b)}\, Id - I_1 \\ e : Id_A(a,b) & \end{array}}{REWR(e, \acute{t}.t(a,b)) : Id_A(a,b)}\, Id - E_1 \quad \rhd_\eta \quad e : Id_A(a,b)$$

This rule is proposed to handle the trivial case, i.e., when the term originated by $a =_t b$ is the simplest one: the term $t(a,b) : Id_A(a,b)$. Since the term e is already a term of type $Id_A(a,b)$, we just reduce $REWR(e, \acute{t}.t(a,b)) : Id_A(a,b)$ directly to e. We chose to call it η-reduction because the η-reduction of the $\lambda\beta\eta$-equality also handles a trivial case, reducing $\lambda x.Mx$ to just M. This rule is the last one of our approach.

3.1 Path-based constructions

The objective of this subsection is to show how to use in practice the rules that we have just defined. The idea is to show construction of terms of some important types. The constructions that we have chosen to build are the reflexive, transitive and symmetric type of the identity type. Those were not random choices. The main reason is the fact that reflexive, transitive and symmetric types are essential to the process of building a groupoid model for the identity type [11]. As we shall see, these constructions come naturally from simple computational paths constructed by the application of axioms of the equality of type theory. In contrast, we will also show that constructing the transitivity using the operator J can be a little complicated. With that, we hope to make the simplicity of our approach clear.

Before we start the constructions, we think that it is essential to understand how to use the eliminations rules. The process of building a term of some type is a matter of finding the right reason. In the case of J, the reason is the correct $x, y : A$ and $z : Id_A(a, b)$ that generates the adequate $C(x, y, z)$. In our approach, the reason is the correct path $a =_g b$ that generates the adequate $h(g) : Id(a, b)$.

One could find strange the fact that we need to prove the reflexivity. Nevertheless, just remember that our approach is not based on the idea that reflexivity is the base of the identity type. As usual in type theory, a proof of something comes down to a construction of a term of a type. In this case, we need to construct a term of type $\Pi_{(a:A)} Id_A(a, a)$. The reason is extremely simple: from a term $a : A$, we obtain the computational path $a =_\rho a : A$:

$$\dfrac{\dfrac{\dfrac{[a : A]}{a =_\rho a : A}}{\rho(a, a) : Id_A(a, a)}\, Id - I_1}{\lambda a.\rho(a, a) : \Pi_{(a:A)} Id_A(a, a)}\, \Pi - I$$

As one can see, we did not need to apply the elimination rule. The reason for that is the fact that the reflexive path has already given us a term of the desired type.

The second proposed construction is the symmetry. We need to construct a term of type $\Pi_{(a:A)}\Pi_{(b:A)}(Id_A(a, b) \to Id_A(b, a))$. As expected, we need to find a suitable reason. Starting from $a =_t b$, we could look at the axioms of *definition 2.3* to plan our next step. One of those axioms makes the symmetry clear: the σ axiom. If we apply σ, we will obtain $b =_{\sigma(t)} a$. Now, it is just a matter of applying the elimination:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[p : Id_A(a, b)] \qquad \dfrac{\dfrac{[a =_t b : A]}{b =_{\sigma(t)} a : A}}{(\sigma(t))(b, a) : Id_A(b, a)}\, Id - I_1}{REWR(p, \acute{t}.(\sigma(t))(b, a)) : Id_A(b, a)}\, Id - E_1}{\lambda p.REWR(p, \acute{t}.(\sigma(t))(b, a)) : Id_A(a, b) \to Id_A(b, a)}\, \Pi - I}{\lambda b.\lambda p.REWR(p, \acute{t}.(\sigma(t))(b, a)) : \Pi_{(b:A)}(Id_A(a, b) \to Id_A(b, a))}\, \Pi - I}{\lambda a.\lambda b.\lambda p.REWR(p, \acute{t}.(\sigma(t))(b, a)) : \Pi_{(a:A)}\Pi_{(b:A)}(Id_A(a, b) \to Id_A(b, a))}\, \Pi - I$$

The third and last construction will be the transitivity. The transitivity will be a special case, since we will also show the construction using the pathless approach, i.e., using the operator J. The objective is to show the difference of complexity in the process of finding a suitable reason. Both approaches have the objective of constructing a term for the type $\Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a, b) \to Id_A(b, c) \to Id_A(a, c))$.

Let's start with the construction based on J. The proof will be based on the one found in [1]. The difference is that instead of defining induction principles for J based on the elimination rules, we will use the rule directly. The complexity is the same, since the proofs are two forms of presenting the same thing and they share

the same reasons. As one should expect, the first and main step is to find a suitable reason. In other words, we need to find suitable $x, y : A$ and $z : Id_A(a, b)$ to construct an adequate $C(x, y, z)$. This first step is already problematic. Different from our approach, where one starts from a path and applies intuitive equality axioms to find a suitable reason, there is no clear point of how one should prooceed to find a suitable reason for the construction based on J. In this case, one should rely on intuition and make attempts until one finds out the correct reason. As one can check in [1], a suitable reason would be $x : A, y : A, - : Id_A(x, y)$ and $C(x, y, z) \equiv Id_A(y, c) \to Id_A(x, c)$. The symbol $-$ indicates that z can be anything, i.e., the choice of z will not matter. Looking closely, the proof is not over yet. The problem is the type of $C(x, x, r(x))$. With this reason, we have that $C(x, x, r(x)) \equiv Id_A(x, c) \to Id_A(x, c)$. Therefore, we cannot assume that $Q(x) : Id_A(x, c) \to Id_A(x, c)$ is the term $r(x)$. The only way to proceed is to apply again the constructor J to build the term $Q(x)$. It means, of course, that we will need to find yet another reason to build this type. This second reason is given by $x : A, y : A, - : Id_A(x, y)$ and $C'(x, y, z) \equiv Id_A(x, y)$. In that case, $C'(x, x, r(x)) = Id_A(x, x)$. We will not need to use J again, since now we have that $r(x) : Id_A(x, x)$. Then, we can construct $Q(x)$:

$$\dfrac{\dfrac{x : A \quad c : A \quad q : Id_A(x, c) \quad r(x) : Id_A(x, x) \quad [x : A] \quad [x : A, y : A, - : Id_A(x, y)] \quad Id_A(x, y)\ \text{type}}{J(q, r(x)) : Id_A(x, c)}\, Id - E}{\lambda q.J(q, r(x)) : Id_A(x, c) \to Id_A(x, c)}$$

Since we have constructed a term to act as the $Q(x)$ of the elimination rule, we can finally obtain the desired term:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a : A \quad b : A \quad p : Id_A(a, b) \quad \lambda q.J(q, r(x)) : Id_A(x, c) \to Id_A(x, c) \quad [x : A] \quad [x : A, y : A, - : Id_A(x, y)] \quad Id_A(y, c) \to Id_A(x, c)\ \text{type}}{J(p, \lambda q.J(q, r(x))) : Id_A(b, c) \to Id_A(a, c)}\, Id - E}{\lambda p.J(p, \lambda q.J(q, r(x))) : Id_A(a, b) \to Id_A(b, c) \to Id_A(a, c)}\, \Pi - I}{\lambda c.\lambda p.J(p, \lambda q.J(q, r(x))) : \Pi_{(c:A)}(Id_A(a, b) \to Id_A(b, c) \to Id_A(a, c))}\, \Pi - I}{\lambda b.\lambda c.\lambda p.J(p, \lambda q.J(q, r(x))) : \Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a, b) \to Id_A(b, c) \to Id_A(a, c))}\, \Pi - I}{\lambda a.\lambda b.\lambda c.\lambda p.J(p, \lambda q.J(q, r(x))) : \Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a, b) \to Id_A(b, c) \to Id_A(a, c))}\, \Pi - I$$

This construction is an example that makes clear the difficulties of working with the pathless model. We had to find two different reasons and use two applications of the elimination rule. Another problem is the fact that the reasons were not obtained by a fixed process, like the applications of axioms in some entity of type theory. They were obtained purely by the intuition that a certain $C(x, y, z)$ should be capable of constructing the desired term. For that reason, obtaining these reasons can be troublesome.

We finish our constructions by giving the path-based construction of the transitivity. The first step, as expected, is to find the reason. Since we are trying to construct the transitivity, it is natural to think that we should start with paths $a =_t b$ and $b =_u c$ and then, from these paths, we shoud conclude that there is a path z that estabilishes that $a =_z c$. To obtain z, we could try to apply the axioms

of *definition 2.3*. Looking at the axioms, one is exactly what we want: the axiom τ. If we apply τ to $a =_t b$ and $b =_u c$, we will obtain a new path $\tau(t,u)$ such that $a =_{\tau(t,u)} c$. Using that construction as the reason, we obtain the following term:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[a =_t b] \quad [b =_u c]}{a =_{\tau(t,u)} c}}{(\tau(t,u))(a,c) : Id_A(a,c)}\, Id-I_1 \quad s(b,c) : Id_A(b,c)}{REWR(s(b,c), \acute{u}(\tau(t,u))(a,c)) : Id_A(a,c)}\, Id-E_1 \quad [w(a,b) : Id_A(a,b)]}{REWR(w(a,b), \acute{t}REWR(s(b,c), \acute{u}(\tau(t,u))(a,c))) : Id_A(a,c)}\, Id-E_1}{\lambda s.REWR(w(a,b), \acute{t}REWR(s(b,c), \acute{u}(\tau(t,u))(a,c))) : Id_A(b,c) \to Id_A(a,c)}}{\lambda w.\lambda s.REWR(w(a,b), \acute{t}REWR(s(b,c), \acute{u}(\tau(t,u))(a,c))) : Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c)}\, \Pi - I}{\lambda c.\lambda w.\lambda s.REWR(w(a,b), \acute{t}REWR(s(b,c), \acute{u}(\tau(t,u))(a,c))) : \Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))}\, \Pi - I}{\lambda b.\lambda c.\lambda w.\lambda s.REWR(w(a,b), \acute{t}REWR(s(b,c), \acute{u}(\tau(t,u))(a,c))) : \Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))}\, \Pi - I$$

$$\lambda a.\lambda b.\lambda c.\lambda w.\lambda s.REWR(w(a,b), \acute{t}REWR(s(b,c), \acute{u}(\tau(t,u))(a,c))) : \Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))\, \Pi - I$$

As one can see, each step is just straightfoward applications of introduction, elimination rules and abstractions. The only idea behind this construction is just the simple fact that the axiom τ guarantees the transitivity of paths. If one compare the reason of this construction to the one that used J, one can clearly conclude that the reason of the path-based approach was obtained more naturally.

4 The Groupoid Model

One of the milestones of the study of the usual identity type was the discovery that the identity type induces an algebraic structure. The structure that it induces is known as groupoid and this connection was originally proposed by Hofmann & Streicher (1994) [11]. As proposed in [11], the identity type obeys the groupoid laws in a special sense: the equations hold up only to propositional equality. Since all the terms that forms the groupoid were constructed using the operator J, the proof that the usual identity type induces a groupoid is not valid for our path-based approach. With that in mind, the objective of this section is to show that our path-based identity type also induces a groupoid structure. Furthermore, our groupoid laws will also hold up only to propositional equality.

Before we show the groupoid laws, understanding the existence of a path between paths will be crucial. Since computational paths are terms of a type (terms of the identity type specifically), all the axioms of *definition 2.3* apply to any path. Then, what will be a path between paths? This is better explained with a simple example. Consider a generic path $a =_s b : A$ and its term $s(a,b) : Id_A(a,b)$. Looking at the axioms of *definition 2.3*, one can apply the axiom ρ in s, obtaining $s =_\rho s$. This path will be the term $\rho(s,s) : Id_{Id_A(a,b)}(s,s)$. That way, $\rho(s,s)$ is a simple example of paths between paths. We are interested in more important examples. Sometimes a path has some redundancies and this redundancies can be reduced, resulting in a new path. Since this new path is just the old one without redundancies, they should be considered propositionally equal, i.e., there should be a path connection these two. Again, it is explained better with some examples:

Example 4.1 Consider a path $a =_t b : A$. It is possible to apply the axiom σ, obtaining $b =_{\sigma(t)} a$ in the process. If we apply again σ, we will obtain $a =_{\sigma(\sigma(t))} b$.

Since we have just inverted the path two times in a row, the path $\sigma(\sigma(t))$ is just a redundant form of the path t. Then, $\sigma(\sigma(t))$ should be reduced (or rewrited) to t, i.e., there should exist a path estabilishing that $\sigma(\sigma(t))$ and t are propositionally equal.

Example 4.2 Consider the reflexive path $a =_\rho a : A$. We can apply σ, obtaining $a =_{\sigma(\rho)} a$. Since we applied σ in a trivial reflexive path, $\sigma(\rho)$ is just a redundant form of ρ. Then, $\sigma(\rho)$ should be reduced to ρ and there should be a path between these two paths.

Example 4.3 Consider the path $a =_t b : A$. If we apply σ, we will obtain $b =_{\sigma(t)} a$. We can take now both t and $\sigma(t)$ and apply the axiom τ, obtaining $a =_{\tau(t,\sigma(t))} a$. Since we have just applied the transitivity to a path and its inverse, the result can be reduced to the reflexivity ρ.

The examples show three distinct cases which redundant paths can be reduced to a path without redundancies. We have also concluded that, in each case, there sould be a path estabilishing the equality of the redundant path and the equivalent one free of redundancies. These examples were simple ones involving only the axioms τ, σ and ρ. Since the equality theory of type theory has a total of seven axioms, as one can check in *definition 2.3*, the combination of these axioms should cause the appearance of many different redundancies. Since we have just showed the simple ones, mapping all these redundancies would be a hard task. Fortunately, this mapping has already been made by De Oliveira (1995) [2].

The analysis of the redundancies generated by the axioms of equality was the main proposal of [2]. In that work, De Oliveira identifies all the redundancies cases and creates rules to reduce these redundancies to a term free of redundancies. These set of all reduction rules forms a system called $LND_{EQ} - TRS$. As we have just seen in the examples, each reduction rules estabilishes that one path (the redundant one) can be rewrited into another one without redundancies, i.e., there is a path between these two paths. The full set of rules of $LND_{EQ} - TRS$ has a total of 39 reduction rules [2]. (In [5], identifiers for each of those definitional equalities were added, so that while the 1st level identifiers for definitional equalities β, η, ξ, etc., are the basis for computational paths of this level, identifiers such as tt, ss, sr, etc., will serve the same role for the next level up.) Fortunately, we are only interested in a subset of these rules, namely the ones that involve the axioms of transitivity, reflexivity and symmetry. Each rule has a name and for the sake of simplicity, the path that the rule induces will receive the same name. Here follow all the rules that interest us and their deductions [2]:

- Rules involving σ and ρ

$$\frac{x =_\rho x : A}{x =_{\sigma(\rho)} x : A} \quad \rhd_{sr} \quad x =_\rho x : A$$

$$\frac{\dfrac{x =_r y : A}{y =_{\sigma(r)} x : A}}{x =_{\sigma(\sigma(r))} y : A} \quad \rhd_{ss} \quad x =_r y : A$$

From these reductions, we obtain the following paths:
$\sigma(\rho) =_{sr} \rho$
$\sigma(\sigma(r)) =_{ss} r$

- Rules involving τ

$$\frac{x =_r y : A \qquad y =_{\sigma(r)} x : A}{x =_{\tau(r,\sigma(r))} x : A} \rhd_{tr} \quad x =_\rho x : A$$

$$\frac{y =_{\sigma(r)} x : A \qquad x =_r y : A}{y =_{\tau(\sigma(r),r)} y : A} \rhd_{tsr} \quad y =_\rho y : A$$

$$\frac{x =_r y : A \qquad y =_\rho y : A}{x =_{\tau(r,\rho)} y : A} \rhd_{trr} \quad x =_r y : A$$

$$\frac{x =_\rho x : A \qquad x =_r y : A}{x =_{\tau(\rho,r)} y : A} \rhd_{tlr} \quad x =_r y : A$$

Obtained paths:
$\tau(r, \sigma(r)) =_{tr} \rho$
$\tau(\sigma(r), r) =_{tsr} \rho$
$\tau(r, \rho) =_{trr} r$
$\tau(\rho, r) =_{tlr} r$

- Rule involving τ and τ

$$\frac{\dfrac{x =_t y : A \qquad y =_r w : A}{x =_{\tau(t,r)} w : A} \qquad w =_s z : A}{x =_{\tau(\tau(t,r),s)} z : A}$$

$$\rhd_{tt} \frac{x =_t y : A \qquad \dfrac{y =_r w : A \qquad w =_s z : A}{y =_{\tau(r,s)} z : A}}{x =_{\tau(t,\tau(r,s))} z : A}$$

Obtained path:
$\tau(\tau(t, r), s) =_{tt} \tau(t, \tau(r, s))$

These paths are all what we need to show that our path-based identity type induces a groupoid structure. Before we show the groupoid laws and the construction of the terms, we'd like to add that the $LND_{EQ}-TRS$ system of rules is terminating and confluent. Terminating in the sense that from a generic path s one can apply the rules until one obtain a path free of redundancies. Confluente in the sense that if a path s can be reduced to m and the same s can also be reduced to n, then there is a r such that m and n can be reduced to r. The proof of these two properties can be found in [2,3,4,7].

4.1 The Groupoid Laws

The groupod model of a type is the idea that the identity type induces a groupoid structure on every type A. Originally proposed by [11], the model is based on the fact that J is capable of constructing terms for the following types [11]:

- $Id_{Id_A(x,z)}(trans(trans(p,q),r), trans(p, trans(q,r)))$
- $Id_{Id_A(x,y)}(trans(refl(x),r),r)$
- $Id_{Id_A(x,y)}(trans(r,refl(x)),r)$
- $Id_{Id_A(y,y)}(trans(symm(r),r),refl(x))$
- $Id_{Id_A(x,x)}(trans(r,symm(r)),refl(x))$
- $Id_{Id_A(x,y)}(symm(symm(r)),r)$

One important detail is, as one can see, that these equalities do not hold "on the nose", so to speak. They hold only up to propositional equality, since every term is an inhabitant of an identity type.

The terms $trans$, $symm$ and $refl$, as one can expect, represent the transitivity, symmetry and reflexivity. (To be precise, only $refl$ is part of the language of terms of identity types in the traditional approach, which leaves open the question as to where do $symm$ and $trans$ come from.) In our path-based interpretation, we write $trans$ as τ, $symm$ as σ and $refl$ as ρ, and they are all identifiers for definitional equalities. Since we already have obtained all the paths we need through the reduction rules, our path-based identity type can construct these terms in a rather natural fashion. For example, consider the path tt between paths, i.e. $\tau(\tau(p,q),r) =_{tt} \tau(p,\tau(q,r))$, then we have:

$$\frac{\tau(\tau(p,q),r) =_{tt} \tau(p,\tau(q,r)) : Id_A(x,z)}{tt((\tau(\tau(p,q),r),\tau(p,\tau(q,r))) : Id_{Id_A(x,z)}(((\tau(\tau(p,q),r),\tau(p,\tau(q,r)))} \; Id - I_1$$

In an analogous way, we can obtain the remaining terms. Hence, we will have all the terms we need:

- $tt((\tau(\tau(p,q),r),\tau(p,\tau(q,r))) : Id_{Id_A(x,z)}(((\tau(\tau(p,q),r),\tau(p,\tau(q,r)))$
- $tlr(\tau(\rho,r),r) : Id_{Id_A(x,y)}(\tau(\rho,r),r)$
- $trr(\tau(r,\rho),r) : Id_{Id_A(x,y)}(\tau(r,\rho),r)$
- $tsr(\tau(\sigma(r),r),\rho) : Id_{Id_A(y,y)}(\tau(\sigma(r),r),\rho)$
- $tr(\tau(r,\sigma(r)),\rho) : Id_{Id_A(x,x)}(\tau(r,\sigma(r)),\rho)$
- $ss(\sigma(\sigma(r)),r) : Id_{Id_A(x,y)}(\sigma(\sigma(r)),r)$

Since paths only estabilish propositional equality, this will be the same case of the pathless approach: all equalities will hold up to propositional equality. We conclude that, for any type A, our path-based identity type induces a groupoid model.

4.2 Sketch of a Higher Structure

In this subsection, our objective is to imply that computational paths can induce a higher structure. We will not expose deeper details, since most of what will be presented in this subsection is still work in progress. Nevertheless, we think that what will be exposed here is interesting enough to be part of this work.

The groupoid model that we have just exposed was based on the fundamental idea that a computational path, as term of a type, is also a computational object. Thefore, it is reasonable to think of paths between paths. Based on that, we can extend the idea and think about paths between paths of paths. We can take it even more further, thinking about paths between paths of paths of paths. In fact, we can be more generic and define the level of a path:

Definition 4.4 The *level* of a path has the following inductive definition:

- If $a, b : A$ are terms such that a and b are not computational paths, then we say that a path $a =_s b$ has level 0.
- If $r, s : A$ are paths of level n, then a path $r =_\theta s$ has level $n + 1$.

The idea is that if we have two n-level paths r and s and if r can be rewrited into s by a sequence of rewrites (a rewrite in the same sense that we defined before), then this sequence of rewrites is a $(n + 1)$-level path that estabilishes that r and s are propositionally equal. Consider now the following structure [12]:

Definition 4.5 Let $n \in \mathbb{N}$. An n-globular set X is the following diagram of sets and functions:

$$X(n) \underset{t}{\overset{s}{\rightrightarrows}} X(n-1) \underset{t}{\overset{s}{\rightrightarrows}} \quad \cdots \quad \underset{t}{\overset{s}{\rightrightarrows}} X(0)$$

The diagram has to obey the following equations for all $x \in \{2, ..., n\}$ and $x \in X(m)$:

$$s(s(x)) = s(t(x)), \quad t(s(x)) = t(t(x))$$

Computational paths can form a globular-set structure. To see this, consider $X(n)$ as the set of n-level paths between two $(n-1)$-level paths (for $n = 0$, just consider two non-path objects) and for any path $x =_m y$, consider that $s(m) = x$ and $t(m) = y$. That way, if $n \geq 2$ and $m \in X(n)$, then $s(m) \in X(n-1)$ and $t(m) \in X(n-1)$. Now, if we consider two n-level paths $t =_\theta m$, and $t =_\alpha m$ and a $(n+1)$-level path $\theta =_\phi \alpha$, then $s(s(\phi)) = s(\theta) = t = s(\alpha) = s(t(\phi))$ and $t(s(\phi)) = t(\theta) = m = t(\alpha) = t(s(\phi))$. All globular-set conditions hold. Then, we can think that computational paths form a globular set structure all up to infinity, i.e., a ∞-globular-set.

Now that we already know that paths form a globular-set structure, we would like to study the redundancies caused by a n-level path. We have exposed the existence of a system known as $LND_{EQ} - TRS$, which estabilishes all rules that solves redundancies between 0-level paths. Since there are redundancies between 0-level paths, it is reasonable to think that there will also exist redundancies between

n-level ones. In a analogous way, we could define a $LND_{EQ} - TRS_n$ system which have all the rules that resolve redundancies between n-level paths. Neverless, the complete understanding of n-level redundancies and what possible rules $LND_{EQ} - TRS_n$ should have is still work in progress. Nonetheless, we hope to achieve some important results involving higher structures. Our hopes are based on the following analogy: if the study of 0-level paths and the $LND_{EQ}-TRS$ resulted in the fact that our path approach induces a groupoid, result that is on par with known results for the traditional identity type, then we strongly believe that our multilevel approach will induce a higher structure similar to the one induced by the pathless identity type. We are refering to the results obtained in [13,15], which concluded that the traditional identity type induces a higher structure known as weak ω-groupoid. Given the high complexity of such structure, obtaining it using our approach will possibly be the main focus of a future work.

5 Conclusion

Inspired by a recent discovery that the propositional equality between terms can be interpreted as a homotopical path, we have revisited the formulation of the intensional identity type, proposing a new approach based on a entity known as *computational path*. We have proposed that a computational path $a =_s b : A$ is a term $s(a, b)$ of the identity type, i.e., $s(a, b) : Id_A(a, b)$, and is formed by a composition of basic rewrites, each with their identifiers taken as constants. We have also developed our approach, showing how the path-based identity type can be constructed and used. In particular, we have showed the simplificity of our elimination rule, showing that it is based on path constructions, which are built from applications of simple axioms of the equality for type theory. To make our point even clearer, we have exposed three path-based constructions. More specifically, constructions that prove the transitivity, reflexivity and symmetry of the identity type. We have also argued that, in our approach, the process of finding the reason that builds the desired term is usually simple and straightfoward. At the same time, in the traditional (or pathless) approach, this is not entirely true, since finding the correct reason can be a cumbersome process.

After estabilishing the foundations of our approach, we analyzed one important structure that the traditional identity type induces: the algebraic structure known as grupoid. Our objective was to show that our approach is on par with the pathless one, i.e., our path-based identity also induces a groupoid sturcture.. To prove that, we have showed that the axioms of equality generate redundancies, which are resolved by paths between paths. We have also exposed that there already exists a system, called $LND_{EQ}-TRS$, that maps those redundancies and proposes rules to reduce them. Using some of these reduction rules, we have obtained all the necessary paths that build the groupoid terms. After obtaining this result, we made a brief sketch of a possible higher structure induced by the path-based identity type. Using paths and deeper level of paths, we have showed that it is possible to form a ∞-globular-set. With that, we have laid the groundwork to prove that our path-based identity type induces a weak ω-groupoid structure.

References

[1] *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, Univalent Foundations Program, Institute for Advanced Study, Princeton, 2014. http://homotopytypetheory.org/book/.

[2] A. G. de Oliveira. *Proof transformations for labelled natural deduction via term rewriting*. Masters thesis, Depto. de Informática, Universidade Federal de Pernambuco, Recife, Brazil, April 1995.

[3] A. G. de Oliveira and R. J. G. B. de Queiroz. Term rewriting systems with labelled deductive systems. In *Proceedings of Brazilian Symposium on Artificial Intelligence (SBIA94)*, pages 59–72, 1994.

[4] A. G. de Oliveira and R. J. G. B. de Queiroz. A normalization procedure for the equational fragment of labelled natural deduction. *Logic Journal of IGPL*, 7(2):173–215, 1999.

[5] R. J. G. B. de Queiroz and A. G. de Oliveira. Propositional equality, identity types, and direct computational paths. 2013. http://arxiv.org/abs/1107.1901.

[6] R. J. G. B. de Queiroz and A. G. de Oliveira. Natural deduction for equality: The missing entity. In *Advances in Natural Deduction*, pages 63–91. Springer, 2014.

[7] R. J. G. B. de Queiroz, A. G. de Oliveira, and D. M. Gabbay. *The Functional Interpretation of Logical Deduction*. World Scientific, 2011.

[8] R. J. G. B. de Queiroz and D. M. Gabbay. Equality in labelled deductive systems and the functional interpretation of propositional equality. In *Proceedings of the 9th Amsterdam Colloquium*, pages 547–565. ILLC/Department of Philosophy, University of Amsterdam, 1994.

[9] R. Harper. Type theory foundations, 2012. Type Theory Foundations, Lecture at Oregon Programming Languages Summer School, Eugene, Oregon.

[10] J. Roger Hindley and Jonathan P. Seldin. *Lambda-calculus and combinators: an introduction*. Cambridge University Press, 2008.

[11] Martin Hofmann and Thomas Streicher. The groupoid model refutes uniqueness of identity proofs. In *Logic in Computer Science, 1994. LICS'94. Proceedings., Symposium on*, pages 208–212. IEEE, 1994.

[12] Tom Leinster. *Higher Operads, Higher Categories*. London Mathematical Society Lecture Note Series (Book 298). Cambridge University Press, 2004. Also http://arxiv.org/abs/math/0305049, May, 2003.

[13] Peter LeFanu Lumsdaine. Weak ω-categories from intensional type theory. In *Typed lambda calculi and applications*, pages 172–187. Springer, 2009.

[14] Dag Prawitz. Inference and knowledge. In *The Logica Yearbook 2008*, pages 175–192. College Publications, London, 2009.

[15] Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.

[16] V. Voevodsky. Univalent foundations and set theory. Univalent Foundations and Set Theory, Lecture at IAS, Princeton, New Jersey, Mar 2014.